Sabrina Oh
Nikolaas Yu
CS302. Data Structures

Project Design Document

- Graph class
    - This class will be templated
        - Template <typename T>
    - Behaviors
        - Constructor (default, parameterized, copy)
          Destructor
    - Members
        - Number of vertices (int)
            - Corresponds to a city
        - List of adjacent cities (list<T>*)

UML Notation	Functionality
getVertices() : T	Returns the number of vertices in the graph
addEdge(start: T, end: T,length: T) : void	Adds an edge between the start and end vertices
printGraph(array[][] : T, numOfCities : T) : void	Takes in the number of cities and the adjacency matrix and outputs each city and the cities that it is adjacent to
setVertices (numVertices : T) : void	Takes in a value of type T, and sets the value of number of vertices to the value passed in

- Graph Functionality Class
    - Attributes

UML Notation	Functionality
isEmpty(theGraph : Graph) : bool	Returns true if graph empty, false if not
getNumVertices(theGraph : Graph) : int	Returns the number of vertices in a graph
getNumEdges(CityClass city) : int	Returns the number of edges to a city
add(T start, T end, T edgeLength): void	If the city name does not exist create a vertex

Remove(T start, T end) : bool	Returns true if the item is removed, otherwise false
getEdgeLength(T start, T end): int	Returns the length of the edge
depthFirstTraversal (T start, void visit(T&))	Should have a public and a protected method. The function listed here is the protected method. The private method will only take in a start. This traverses the graph starting at the start specified, in a depth first manner. It will output the output the order of cities visited
breadthFirstSearch(T start, void visit(ItemType&))	Should have a public and a private function. The function listed here is the protected method. This function takes in a start location, and traverses the graph in a breadth first manner. It will output the order of cities visited

- Graph Interface Class
    - This interface is templated, but all the functions it contains are pure virtual functions
    - Inherits from Graph Class

UML Notation	Functionality
virtual int getNumVertices() const =0	
virtual int getNumEdges() const = 0	
virtual bool add (ItemType start, ItemType end, int edgeLength) = 0	
virtual bool remove (ItemType start, ItemType end) const = 0	
virtual int getEdgeLength (ItemType start, ItemType end) const = 0	
virtual void depthFirstTraversal (ItemType start, void visit (ItemType&))	
virtual void breadthFirstTraversal (ItemType start, void visit (ItemType&))	

virtual ~GraphInterface() {}	

- findList Class
    - Will inherit from the list class in the Standard Template Library
    - Used to store cities already visited

find(T Item):bool	Returns True if item occurs in the list

- Adjacency Matrix
    - Numbers in parenthesis is approximate driving distance (edge weight)

	Reno	San Francisco	Salt Lake City	Seattle	Las Vegas
Reno	0	218	518	704	443
San Francisco	218	0	0	808	564
Salt Lake City	518	0	0	840	425
Seattle	704	808	840	0	0
Las Vegas	443	564	425	0	0

    - When array is passed into program
        - 0: Reno
        - 1: San Francisco
        - 2: Salt Lake City
        - 3: Seattle
        - 4: Las Vegas
- Possible Combinations and Weights
    - See Excel spreadsheet for different combinations and weights

Main Driver

- Functionality
- Outside of the main function
    - A templated visit function, return type void
        - Takes in a templated item, and prints to the screen the item
- Input a 2D array that corresponds to the adjacency matrix
- Will use a modified shortestPath algorithm that accounts for cities visited.
- Current cost of gas
    - Ask user

Responsibilities

We will both be contributing to the main driver as our classes will require for complete functionality. As the Graph Functionality class is considerably more work and the

GraphInterface class contains no implementation details, we will also collaborate on implementing the more difficult function of the class. These include the breadth first traversal, depth first traversal, and a minimum path algorithm.

Sabrina

- Brute Force Calculations Spreadsheet
- Graph Class
    - Creates a graph
    - Holds the number of vertices
    - Adds edges based on adjacency matrix
    - Outputs the edges adjacent to each vertex
- GraphInterface Class
    - Contains a class called GraphInterface
    - Consists only of pure virtual functions

Nikolaas

- Graph Functionality Class
    - Contains a class for graph functionalities such as search algorithms
- FindList

Timeline

April 30: Graph Interface Class Completed

May 2: Graph Class Completed, FindList Completed

May 4: Graph Functionality Class Completed

May 6: Project Deadline

Communication

- Will communicate primarily through Discord
- Phone contact information has been exchanged in the event we are not able to contact through Discord

Technologies Used

- We will be utilizing GitHub, and our respective text editors